IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application. No: 10/649,903	§	Atty. Dkt. No: 5150-77400
Filed: August 26, 2003	§	
Inventor(s): Mahesh A. Ramchandani	§	
Title: BINDING A GUI ELEMENT TO A CONTROL IN A TEST EXECUTIVE APPLICATION	§	
Examiner: Mitchell, Jason D.	§	
Group/Art Unit: 2193	§	

APPEAL BRIEF

Box: Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir/Madam:

Further to the Notice of Appeal filed December 15, 2008, Appellant presents this Appeal Brief. Appellant respectfully requests that this appeal be considered by the Board of Patent Appeals and Interferences.

I. REAL PARTY IN INTEREST

The subject application is owned by National Instruments Corporation, a corporation organized and existing under and by virtue of the laws of the State of Delaware, and having its principal place of business at 11500 N. MoPac Expressway, Bldg. B, Austin, Texas 78759-3504.

II. RELATED APPEALS AND INTERFERENCES

No related appeals or interferences are known which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1-75, 87, and 108 are canceled.

Claims 76-86, 88-107, and 109-114 are pending in the application. All of the pending claims stand rejected and are the subject of this appeal. A copy of the claims incorporating entered amendments is included in the Claims Appendix of the previously filed Appeal Brief.

IV. STATUS OF AMENDMENTS

All amendments have been entered. The Claims Appendix hereto reflects the current state of the claims.

V. <u>SUMMARY OF THE INDEPENDENT CLAIMS</u>

Independent claim 76 relates generally to a computer-implemented method for displaying information regarding a test executive sequence, where the test executive sequence includes a plurality of steps. The method comprises including a graphical user interface (GUI) element (*p. 3, lines 10-11*) in a graphical user interface of a run-time operator interface application (*p. 2, lines 12-15*) in response to user input (*FIG. 7, block 327; p. 23, lines 12-14*). The GUI element is operable to display information (*p. 4, lines 17-18*).

The method also comprises including a control (*p. 10, lines 1-2*) in the run-time operator interface application in response to user input (*FIG. 7, block 329; p. 23, lines 18-20*). The control includes pre-existing first functionality for determining the steps in the test executive sequence (*p. 24, lines 17-19*).

The method further comprises configuring a binding between the GUI element and the control (*FIG. 7, block 331; p. 23, lines 26-27*). Configuring the binding enables the GUI element to automatically display at least a subset of the steps in the test executive sequence in response to the control determining the steps in the test executive sequence during execution of the run-time operator interface application (*p. 24, lines 19-23*).

The method further comprises executing the run-time operator interface application (*p. 23, line 30 – p. 24, line 3*). Executing the run-time operator interface application comprises the control executing to automatically determine the steps in the test executive sequence (*p. 23, line 30 – p. 24, line 3*). The binding between the GUI element and the control causes the GUI element to automatically display at least a subset of the steps in response to the control determining the steps (*p. 24, lines 19-23*). The GUI element displays the at least a subset of the steps in the graphical user interface of the run-time operator interface application during execution of the run-time operator interface application (*FIG. 8*).

Independent claim 94 relates generally to a computer-implemented method for displaying a report for a test executive sequence execution. The method comprises including a GUI element (*p. 3, lines 10-11*) in a graphical user interface of a run-time operator interface application (*p. 2, lines 12-15*) in response to user input (*FIG. 7, block 327; p. 23, lines 12-14*). The GUI element is operable to display information (*p. 4, lines 17-18*).

The method also comprises including a control (*p. 10, lines 1-2*) in the run-time operator interface application in response to user input (*FIG. 7, block 329; p. 23, lines 18-20*). The control includes pre-existing first functionality for generating a report summarizing one or more results of execution of the test executive sequence (*p. 24, lines 24-25*).

The method further comprises configuring a binding between the GUI element and the control (*FIG. 7, block 331; p. 23, lines 26-27*). Configuring the binding enables the GUI element to automatically display the report in response to the control generating the report during execution of the run-time operator interface application (*p. 24, lines 25-27*).

The method further comprises configuring the run-time operator interface application to invoke execution of the test executive sequence (*p. 8, lines 29-31; p. 23, lines 4-5*).

The method further comprises executing the run-time operator interface application (*p. 23, line 30 – p. 24, line 3*). The run-time operator interface application executes to invoke execution of the test executive sequence (*p. 8, lines 29-31; p. 23, lines 4-5*). The execution of the test executive sequence produces the one or more results (*p. 17, line 14*). The control automatically generates the report summarizing the one or more results of the execution of the test executive sequence in response to the execution of the test executive sequence (*p. 24, lines 24-25*). The binding between the GUI element and the control causes the report to be automatically displayed by the GUI element in response to the control generating the report (*p. 24, lines 24-31*). The GUI element displays the report in the graphical user interface of the run-time operator interface application during execution of the run-time operator interface application (*p. 24, lines 24-31*).

Independent claim 95 relates generally to a computer-implemented method for displaying information for a test executive sequence execution. The method comprises including a GUI element (*p. 3, lines 10-11*) in a graphical user interface of a run-time operator interface application (*p. 2, lines 12-15*) in response to user input (*FIG. 7, block 327; p. 23, lines 12-14*). The GUI element is operable to display information (*p. 4, lines 17-18*).

The method further comprises including a control (*p. 10, lines 1-2*) in the run-time operator interface application in response to user input (*FIG. 7, block 329; p. 23, lines 18-20*). The control includes pre-existing first functionality for generating information indicating one or more execution results for the test executive sequence (*p. 24, lines 24-25*).

The method further comprises configuring a binding between the GUI element and the control (*FIG. 7, block 331; p. 23, lines 26-27*). Configuring the binding enables the GUI element to automatically display the information in response to the control generating the information during execution of the run-time operator interface application (*p. 24, lines 25-27*).

The method further comprises configuring the run-time operator interface application to invoke execution of the test executive sequence (*p. 8, lines 29-31; p. 23, lines 4-5*).

The method further comprises executing the run-time operator interface application (*p. 23, line 30 – p. 24, line 3*). The run-time operator interface application executes to invoke execution of the test executive sequence (*p. 8, lines 29-31; p. 23, lines 4-5*). The execution of the test executive sequence produces the one or more execution results (*p. 17, line 14*). The control automatically generates the information indicating the one or more execution results in response to the execution of the test executive sequence (*p. 24, lines 24-25*). The binding between the GUI element and the control causes the information indicating the one or more execution results to be automatically displayed by the GUI element in response to the control generating the information (*p. 24, lines 24-31*). The GUI element displays the information in the graphical user interface of the run-time operator interface application during execution of the run-time operator interface application (*p. 24, lines 24-31*).

Independent claim 96 relates generally to a computer-implemented method for creating a run-time operator interface application for controlling execution of a test executive sequence. The method comprises including a GUI element (*p. 3, lines 10-11*) in a graphical user interface of the run-time operator interface application (*p. 2, lines 12-15*) in response to user input (*FIG. 7, block 321; p. 21, lines 16-23*).

The method further comprises including a control (*p. 10, lines 1-2*) in the run-time operator interface application in response to user input (*FIG. 7, block 323; p. 21, lines 24-26*). The control includes pre-existing first functionality for invoking execution of the test executive sequence (*p. 23, lines 4-7*).

The method further comprises configuring a binding between the GUI element and the control (*FIG. 7, block 325; p. 22, lines 3-12*). Configuring the binding enables the control to automatically invoke execution of the test executive sequence in response to user input received to the GUI element during execution of the run-time operator interface application (*p. 22, lines 3-12; p. 23, lines 4-6*).

The method further comprises executing the run-time operator interface application (*p. 22, line 6*). Executing the run-time operator interface application comprises displaying the GUI element in the graphical user interface of the run-time operator interface application and receiving user input to the GUI element during execution of the run-time operator interface application (*p. 22, lines 6-8*). The binding between the GUI element and the control causes the control to automatically invoke execution of the test execution sequence in response to the user input to the GUI element (*p. 22, lines 8-21; p. 23, lines 4-7*).

Independent claim 113 relates generally to a computer-implemented method for creating a run-time operator interface application for controlling execution of a test executive sequence. The method comprises including a GUI element (*p. 3, lines 10-11*) in a graphical user interface of the run-time operator interface application (*p. 2, lines 12-15*) in response to user input (*FIG. 7, block 321; p. 21, lines 16-23*).

The method further comprises including a control (*p. 10, lines 1-2*) in the run-time operator interface application in response to user input (*FIG. 7, block 323; p. 21, lines*

24-26). The control includes pre-existing first functionality for stopping execution of the test executive sequence (*p. 23, lines 7-9*).

The method further comprises configuring a binding between the GUI element and the control (*FIG. 7, block 325; p. 22, lines 3-12*). Configuring the binding enables the control to automatically stop execution of the test executive sequence in response to user input received to the GUI element during execution of the run-time operator interface application (*p. 22, lines 3-12; p. 23, lines 7-9*).

The method further comprises executing the run-time operator interface application (*p. 22, line 6*). Executing the run-time operator interface application comprises displaying the GUI element in the graphical user interface of the run-time operator interface application and receiving user input to the GUI element during execution of the run-time operator interface application (*p. 22, lines 6-8*). The binding between the GUI element and the control causes the control to automatically stop execution of the test execution sequence in response to the user input to the GUI element (*p. 22, lines 8-21; p. 23, lines 7-9*).

Independent claim 114 relates generally to a computer-implemented method for creating a run-time operator interface application for controlling execution of a test executive sequence. The method comprises including a first GUI element (*p. 3, lines 10-11*) in a graphical user interface of the run-time operator interface application (*p. 2, lines 12-15*) in response to user input (*FIG. 7, block 321; p. 21, lines 16-23*). The first GUI element is operable to receive user input during execution of the run-time operator interface application (*p. 21, lines 20-23*).

The method further comprises including a second GUI element in the graphical user interface of the run-time operator interface application in response to user input (*FIG. 7, block 327; p. 23, lines 12-14*). The second GUI element is operable to display output during execution of the run-time operator interface application (*p. 23, lines 12-14*).

The method further comprises including a control (*p. 10, lines 1-2*) in the run-time operator interface application in response to user input (*FIG. 7, block 329; p. 23, lines 18-20*). The control includes pre-existing first functionality (*p. 21, lines 25-26*) for selecting the test executive sequence (*p. 22, lines 22-24*). The control also includes pre-existing second functionality (*p. 22, lines 28-30; p. 23, lines 19-20; p. 25, lines 9-10 and 15-16*) for displaying steps in the test executive sequence (*p. 24, lines 17-19*).

The method further comprises configuring a first binding between the first GUI element and the control (*FIG. 7, block 325; p. 22, lines 3-12*). The first binding enables the control to automatically invoke a dialog box enabling a user to select the test executive sequence in response to user input received to the first GUI element during execution of the run-time operator interface application (*p. 22, lines 3-12 and 22-31*).

The method further comprises configuring a second binding between the second GUI element and the control (*FIG. 7, block 331; p. 23, lines 26-27*). The second binding enables the second GUI element to automatically display the steps in the test executive sequence in response to the user selecting the test executive sequence during execution of the run-time operator interface application (*p. 22, lines 28-30; p. 24, lines 19-23*).

The method further comprises executing the run-time operator interface application (*p. 22, line 6*). Executing the run-time operator interface application comprises displaying the first GUI element and the second GUI element in the graphical user interface of the run-time operator interface application and receiving user input to the first GUI element (*p. 22, lines 6-8*). The control automatically invokes a dialog box enabling a user to select the test executive sequence in response to the user input to the first GUI element (*p. 22, lines 22-28*). User input selecting the test executive sequence is received via the dialog box (*p. 22, lines 22-28*), and the second GUI element automatically displays the steps in the test executive sequence in response to the user input selecting the test executive sequence (*p. 22, lines 28-30*). The second GUI element displays the steps in the graphical user interface of the run-time operator interface application during execution of the run-time operator interface application (*FIG. 8*).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Section 103 Rejections (Grey in view of Stutz)

Claims 76-86, 88-98, 100-107 and 109-114 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Grey et al., U.S. Patent No. 6,401,220 (hereinafter "Grey") in view of Stutz et al., U.S. Patent No. 5,485,617 (hereinafter "Stutz").

Section 103 Rejections (Grey in view of Stutz and Carter)

Claim 91 stands rejected under 35 U.S.C. 103(a) as being unpatentable over Grey in view of Stutz, and further in view of U.S. Patent No. 6,718,534 to Carter et al. (hereinafter "Carter").

VII. ARGUMENT

Section 103 Rejections (Grey in view of Stutz)

Claims 76-86, 88-98, 100-107 and 109-114 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Grey in view of Stutz. Appellant respectfully traverses these rejections.

Independent Claim 76

Grey relates generally to a test executive system that allows a user to create a test sequence for testing a unit under test. FIG. 2 illustrates the components of Grey's system. The system includes a sequence editor 212. The sequence editor is "a program that provides a graphical user interface for creating, editing, and debugging sequences." (Col. 2, lines 3-5). More specifically, the sequence editor allows the user to create a sequence of steps, e.g., where different steps in the sequence call different test modules to perform specific tests on the unit under test. (Col. 1, lines 52-64). The system also includes run-time operator interface programs 202. A run-time operator interface program is "a program that provides a graphical user interface for executing sequences on a production station." (Col. 2, lines 6-8). Thus, a user first creates a test sequence using the sequence editor. The sequence is then executed in a run-time operator interface program. Executing the sequence involves executing each of the steps in the sequence, e.g., executing each of the test modules specified by the steps. (Col. 1, lines 61-63).

The present claims relate specifically to run-time operator interface applications (programs), e.g., applications that provide a graphical user interface that enables a user to control the execution of a test executive sequence. Claim 76 relates recites in pertinent part:

> including a control in the run-time operator interface application in response to user input, wherein the control includes pre-existing first functionality for determining the steps in the test executive sequence;
>
> configuring a binding between the GUI element and the control, wherein configuring the binding enables the GUI element to automatically display at least a subset of the steps in the test executive sequence in response to the control determining the steps in the test executive sequence during execution of the run-time operator interface application;

These limitations relate generally to the creation of the run-time operator interface application. More particularly, these limitations relate to a control that can be included in a run-time operator interface application, where the "control includes pre-existing first functionality for determining the steps in the test executive sequence". The control is bound to a GUI element which is also included in the run-time operator interface application. Configuring the binding between the GUI element and the control enables the GUI element to automatically display at least a subset of the steps in the test executive sequence in response to the control determining the steps in the test executive sequence during execution of the run-time operator interface application.

Grey, taken either singly or in combination with Stutz, does not teach these limitations in combination with the other limitations recited in claim 76. With respect to the limitation of, "including a control in the run-time operator interface application in response to user input, wherein the control includes pre-existing first functionality for determining the steps in the test executive sequence," the Examiner states:

> col. 4, lines 4-5 "The TestStand Engine automatically determines the type being loaded"

However, this portion of Grey has nothing to do with a control that includes pre-existing first functionality for determining the steps in the test executive sequence. The cited portion of Grey instead relates to step <u>types</u>. A step type "essentially comprises a custom set of properties and/or operations associated with a step." The user can define various step types. When a given step is created and added to the sequence, the user can request that the step be of a particular <u>step type</u>. Grey teaches that, "For each type that a file uses, the TestStand system stores the definition of the type in the file." (Col. 3, lines 38-40). The teaching cited by the Examiner refers to the file subsequently being loaded after the step type definitions have been stored in it. Grey teaches here that, "In response to the user loading the file, the TestStand Engine automatically determines the type being loaded, and then automatically determines if the loaded type conflicts with one or more previously loaded/registered types." (Col. 4, lines 4-5). Thus, the cited passage teaches nothing whatsoever about a control that includes pre-existing first functionality for determining <u>the steps</u> (not the step <u>types</u>!) in the test executive sequence.

In response to the arguments presented above, the Examiner states on p. 3 of the Office Action of September 15, 2008:

> The examiner respectfully disagrees. First it is noted that the applicant has failed to indicate the perceived distinction between the claimed determining a "step" and Grey's determining of a "step type" (see col. 3, lines 35-38). Accordingly, the applicant's arguments fail to comply with 37 CFR I .I I I (b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references.

Appellant respectfully disagrees and submits that one skilled in the art would readily recognize the difference between a <u>step</u> and a <u>step type</u> in view of Grey's disclosure. As discussed above, the user can define various step types. When a given step is created and added to the sequence, the user can request that the step be of a particular <u>step type</u>. Thus, the step will automatically be configured with the custom set of properties and/or operations defined for the step type. For example, in FIGs. 16-18 Grey illustrates various tabs of a Step Type Properties dialog box which enables a user to set properties for a Numeric Limit Test step type. After the Numeric Limit Test step type has been defined, the user can then create various instances of Numeric Limit Test steps, i.e., steps of the Numeric Limit Test step type. The steps will automatically inherit the properties that were defined for the Numeric Limit Test step type. Thus, the Numeric Limit Test step type is not itself a step, but is a type which defines properties for steps having that step type.

On p. 4 of the Office Action the Examiner further argues:

> Further, those of ordinary skill in the art would have recognized that determining the 'type' of a step must first include identifying the step who's type is being identified (as well as the "set of properties andlor operations associated with [the] step".

Appellant respectfully disagrees. As noted above, Grey teaches, "For each type that a file uses, the TestStand system stores the definition of the type in the file." (Col. 3, lines 38-40). The teaching cited by the Examiner refers to the file subsequently being loaded after the step type definitions have been stored in it. Grey teaches here that, "In response to the user loading the file, the TestStand Engine automatically determines the <u>type</u> being loaded, and then automatically determines if the loaded <u>type</u> conflicts with one or more previously loaded/registered types." Thus, Grey teaches determining the step types in the file. As discussed above, step types are defined independently of any particular step, and thus the step type definitions that have been stored in the file can be determined without determining the actual steps in a test executive sequence.

The Examiner also argues:

> Additionally, Grey discloses "determining if the name of the loaded type conflicts with the name of any of the previously loaded/registered types" (col. 4, lines 9-1 3). From this it can be seen that Grey's system has determined the "name" of a type and thus provides a control which "determine[s] the steps in the test executive sequence".

Appellant again respectfully disagrees. As discussed above, a step type is different from a step. Determining the name of a step type is not the same as determining the name of a step or "determining the steps in the test executive sequence."

As noted above, claim 76 further recites:

> configuring a binding between the GUI element and the control, wherein configuring the binding enables the GUI element to automatically display at least a subset of the steps in the test executive sequence in response to the control determining the steps in the test executive sequence during execution of the run-time operator interface application;

Regarding these limitations the Examiner cites:

> col. 3, lines 16-18 "The sequence editor ... interface[s] to the test executive engine"

and

> Fig. 4 see the 'Main' tab, 'Step' colum

The limitations in question relate to the "at least a subset of the steps" being automatically displayed in the GUI element during execution of the run-time operator interface application. However, the cited portions of Grey teach nothing whatsoever about any steps of the sequence being displayed during execution of the run-time operator interface application. As per Col. 3, lines 16-18, Grey merely teaches here that:

> The sequence editor and the operator interface programs interface to the test executive engine, referred to as the TestStand Engine. One or more process

> models couple to the TestStand Engine. The TestStand Engine interfaces through an adapter interface to one or more adapters. The adapters in turn interface to code modules and sequence files.

Thus, Grey is here simply describing the architecture of the test executive system, teaching that the sequence editor and the operator interface programs interface to the test executive engine, e.g., as shown in FIG. 2. In contrast, claim 76 recites, "configuring a binding between the GUI element and the control". <u>The cited portions of Grey teach nothing about configuring a binding between a GUI element and a control</u>.

As per the "Main" tab and the "Step" column in FIG. 4 cited by the Examiner, this is a screen shot illustrating Grey's <u>sequence editor</u>. More particularly, the screen shot illustrates the steps in the default process model. (See Col. 21, lines 48-50). A process model is basically a series of steps that are commonly used by different test sequences and acts as a template so that the user does not have to create the same steps every time for different sequences. Grey teaches that, "A process model is in the form of a sequence file. The user can edit a process model just as he/she edits other sequences." (Col. 20, lines 20-22). Thus, the screen shot in FIG. 4 illustrates the sequence editor and relates generally to creating the sequence. In contrast, the recited limitations in claim 76 relate to the "at least a subset of the steps" being automatically displayed <u>during execution of the run-time operator interface application</u> in response to the control determining the steps, <u>where the binding configured between the GUI element and the control enable the GUI element to automatically display the "at least a subset of the steps</u>". Grey simply does not teach this subject matter, taken either singly or in combination with Stutz.

In response to these arguments the Examiner states on p. 5 of the Office Action:

> The examiner respectfully disagrees. First, Grey's fig. 4 shows an <u>operator interface application</u> (i.e. a GUI element of an application) displaying the steps in a test executive sequence (e.g. "Check for Proper Use", "Clear Report", "PreUUTLoop Callback"). Further, as Grey discloses a GUI element of an operator interface application, the data in that GUI element must necessarily be shown at runtime. In other words, if the application is not running the GUI can not be displayed. Further, Grey discloses the GUI element (see Fig. 4) "interfaces" with the control (col. 4, lines 4-7 "the TestStandEngine"). (*Emphasis added*)

Appellant respectfully disagrees. FIG. 4 does not show an <u>operator interface application</u>, as asserted by the Examiner. FIG. 4 instead illustrates Grey's <u>sequence editor</u>, as noted above. As discussed at the beginning of the arguments with respect to claim 76, Grey teaches that the sequence editor is "a program that provides a graphical user interface for <u>creating, editing, and debugging</u> sequences." (Col. 2, lines 3-5). More specifically, the sequence editor allows the user to create a sequence of steps, e.g., where different steps in the sequence call different test modules to perform specific tests on the unit under test. (Col. 1, lines 52-64). A run-time operator interface program is "a program that provides a graphical user interface for <u>executing</u> sequences on a production station." (Col. 2, lines 6-8). Thus, a user first creates a test sequence using the sequence editor. The sequence is then executed in a run-time operator interface program.

Appellant thus re-asserts the argument that the cited portions of Grey (and Grey and general) do not teach the "at least a subset of the steps" being automatically displayed **<u>during execution of the run-time operator interface application</u>** in response to the control determining the steps, <u>where the binding configured between the GUI element and the control enables the GUI element to automatically display the "at least a subset of the steps</u>".

Claim 76 further recites:

> executing the run-time operator interface application, wherein said executing comprises the control executing to automatically determine the steps in the test executive sequence, wherein the binding between the GUI element and the control causes the GUI element to automatically display at least a subset of the steps in response to the control determining the steps, wherein the GUI element displays the at least a subset of the steps in the graphical user interface of the run-time operator interface application during execution of the run-time operator interface application.

With respect to these limitations, the Examiner again cites the same portions of Grey previously discussed above. Thus, Appellant disagrees with the Examiner for similar reasons to those set forth above. For example, with respect to the limitation of, "executing the run-time operator interface application, wherein said executing comprises the control executing to automatically determine the steps in the test executive sequence," the Examiner again cites:

col. 4, lines 4-5 "The TestStand Engine automatically determines the type being loaded"

However, as discussed above, this portion of Grey pertains to the determination of step types which have been stored in a file, not to the determination of steps in a test executive sequence. Furthermore, this portion of Grey is not at all related to the execution of a run-time operator interface application.

With respect to the limitations of "wherein the binding between the GUI element and the control causes the GUI element to automatically display at least a subset of the steps in response to the control determining the steps, wherein the GUI element displays the at least a subset of the steps in the graphical user interface of the run-time operator interface application during execution of the run-time operator interface application," the Examiner again cites Grey at FIG. 4. However, as discussed above, this portion of Grey has nothing to do with the execution of a run-time operator interface application, but instead pertains to Grey's sequence editor.

Furthermore, Appellant also submits that the Examiner has not established a proper motivation to combine Stutz with Grey. The Examiner states:

> It would have been obvious to one of ordinary skill in the art at the time the invention was made to develop the run-time operator interface disclosed in Grey using the methods taught in Stutz (col. 10, lines 46-48) because Stutz provides "an improved method … for dynamically generating object connections (col. 8, lines 14-17).

Stutz relates generally to creating connections between objects in a program. Stutz's invention operates at a fairly low level of programming, e.g., in order to allow a source object to notify a sink object. (See Abstract). In contrast, Grey's invention operates at a much higher level and relates to a test executive system that allows a user to create a test sequence, e.g., through the graphical user interface of a sequence editor such as illustrated in FIG. 4. It is difficult to see the particular relevance of Stutz's invention to Grey's invention or to understand how or why Stutz's teaching of dynamically generating object connections would be incorporated into Grey's system.

With respect to claim 76, and the other independent claims in general, Appellant further notes the following arguments.

Consider the problem at hand: A user has created a test executive sequence and now wishes to create a run-time operator interface application for the sequence. In particular, in the case of claim 76, the user wants the "at least a subset of the steps in the test executive sequence" to be automatically displayed in the GUI element in the run-time operator interface application. The user can configure this to happen automatically by simply configuring the binding between the GUI element and the control that has the pre-existing functionality for determining the steps in the test executive sequence. The specification of the present application teaches that:

> Controls such as those described above may advantageously remove the burden on the user (programmer) from implementing at least a portion of the functionality for a test executive application or run-time operator interface application. For example, a common feature for run-time operator interface applications is to display a list of steps in a test executive sequence. <u>A control such as described above may be operable to automatically display the steps of a specified sequence and may eliminate the need for the user (programmer) to write code to perform such tasks as: obtaining a reference to a sequence file containing the test executive sequence, enumerating and obtaining references to the sequences that are in the sequence file, enumerating and obtaining information regarding steps of the desired sequence, formatting the step information appropriately, displaying the formatted information, etc. Instead, a control bound to a GUI element in the test executive application may automatically perform these tasks.</u> (*p. 30, lines 6-17*)

Thus, by providing a control which is operable to automatically determine the steps and display them in the GUI element which is bound to the control, the user (e.g., the programmer who creates the run-time operator interface application) is freed from the task of having the implement this functionality.

Grey teaches a test executive system which includes a built-in sequence editor and default run-time operator interfaces. The built-in sequence editor is operable to display the steps in a test sequence, e.g., as shown in FIG. 4. However, Grey does not contain any teaching regarding the user of the test executive system creating his own run-time operator interface application by including in the run-time operator interface application a control with pre-existing functionality to automatically determine and display the steps in the sequence in a GUI element, as recited in claim 76. Furthermore, Stutz does not remedy this deficiency of Grey's teaching.

Appellant respectfully reminds the Board that, "when evaluating the scope of a claim, every limitation in the claim must be considered. <u>Office personnel may not dissect a claimed invention into discrete elements and then evaluate the elements in isolation. Instead, the claim as a whole must be considered.</u> See, e.g., *Diamond v. Diehr*, 450 U.S. at 18889, 209 USPQ at 9" (as quoted from the MPEP 2106, Section II, Part C) (emphasis added). When taken as a whole, Grey and Stutz do not teach the subject matter recited in claim 76. The combined references do not teach a control which can be included in a run-time operator interface application in response to user input and which includes pre-existing first functionality for determining the steps in the test executive sequence, where a GUI element can be bound to the control such that the binding enables the GUI element to automatically display at least a subset of the steps in response to the control determining the steps in the test executive sequence during execution of the run-time operator interface application, as recited in claim 76.

In response to these arguments the Examiner states on p. 8 of the Office Action:

> The examiner respectfully disagrees. The claims do not recite "The user of the test executive system creating his own run-time operator interface application by including in the run-time operator interface application a control with pre-existing functionality to automatically determine and display the steps in the sequence in a GUI element". Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See In re Van Geuns, 988 F.2d 11 81, 26 USPQ2d 1057 (Fed. Cir. 1993).

However, as discussed above, claim 76 recites in pertinent part:

> <u>including a control in the run-time operator interface application in response to user input, wherein the control includes pre-existing first functionality for determining the steps in the test executive sequence;</u>
>
> configuring a binding between the GUI element and the control, <u>wherein configuring the binding enables the GUI element to automatically display at least a subset of the steps in the test executive sequence in response to the control determining the steps in the test executive sequence during execution of the run-time operator interface application</u>; and
>
> executing the run-time operator interface application, wherein said executing comprises <u>the control executing to automatically determine the steps in the test executive sequence, wherein the binding between the GUI element and the control causes the GUI element to automatically display at least a subset</u>

> <u>of the steps in response to the control determining the steps, wherein the GUI element displays the at least a subset of the steps in the graphical user interface of the run-time operator interface application during execution of the run-time operator interface application</u>.

Applicant respectfully submits that it would be clear to one skilled in the art (particularly in view of Applicant's specification) that including a control having the recited pre-existing functionality in the run-time operator interface application in response to user input would aid a user in solving the problem described above, whereas Grey and Stutz provide no hint of any functionality for solving the described problem. Furthermore, Applicant respectfully submits that Grey and Stutz do not teach the recited combination of limitations, as argued above.

Appellant thus respectfully submits that claim 76, and the claims dependent thereon, are patentably distinct over Grey and Stutz for at least the reasons set forth above.

<u>Dependent Claims 77 and 78</u>

Claim 77 recites the further limitations of:

> wherein the control also includes pre-existing functionality for formatting the at least a subset of the steps in the test executive sequence into a formatted list;
>
> wherein the GUI element automatically displaying the at least a subset of the steps comprises the GUI element automatically displaying the formatted list.

In the rejection of claim 77, the Examiner cites Grey's teaching at Col. 31, lines 31-33:

> The General tab is used to specify a name, description, and comment for the step type. The user also can specify an icon and a module adapter.

This pertains to the user defining a <u>step type</u> such as described above and teaches nothing whatsoever about a control that can be included in a run-time operator interface application, where the control includes pre-existing functionality for <u>formatting the at least a subset of the steps into a formatted list</u>.

Appellant thus respectfully submits that claim 77, and claim 78 which recites additional limitations regarding the formatting, are separately patentable over Grey and Stutz for at least the reasons set forth above.

<u>Dependent Claims 80 and 81</u>

Claim 80 recites the further limitations of:

> configuring the control in response to configuration user input after said including the control in the run-time operator interface application, wherein the configuration user input specifies an appearance for the displayed steps, wherein configuring the control enables the control to cause the steps to be displayed in the GUI element with the specified appearance.

With respect to these limitations the Examiner cites Grey's teaching at Col. 31, lines 31-33:

> The General tab is used to specify a name, description, and comment for the step type. The user also can specify an icon and a module adapter.

As discussed above, this pertains to the user defining a name, description, and comment for a <u>step type</u>. Grey teaches nothing whatsoever about configuring a control in response to configuration user input that specifies an appearance for the steps that are displayed in the GUI element, where configuring the control enables the control to cause the steps to be displayed in the GUI element with the specified appearance.

The Examiner cites the same teaching in the rejection of claim 81. However, Grey teaches nothing regarding the specific limitations of, "wherein the configuration user input specifies one or more properties regarding a plurality of columns to display in the GUI element" and "wherein configuring the control enables the control to cause information for each displayed step to be displayed in the GUI element in the plurality of columns according to the one or more specified properties."

Appellant thus respectfully submits that claims 80 and 81 are separately patentable over Grey and Stutz for at least the reasons set forth above.

<u>Dependent Claim 82</u>

Claim 82 recites the further limitations of:

wherein the GUI element comprises a first GUI element;

wherein the method further comprises:

including a second GUI element in the run-time operator interface application in response to user input; and

configuring a binding between the second GUI element and the control;

wherein executing the run-time operator interface application comprises the second GUI element receiving user input during execution of the run-time operator interface application;

wherein the binding between the second GUI element and the control causes the control to automatically determine the steps in the test executive sequence in response to the user input received to the second GUI element during execution of the run-time operator interface application;

wherein the binding between the first GUI element and the control causes the first GUI element to automatically display the at least a subset of the steps in response to the user input received to the second GUI element during execution of the run-time operator interface application.

Thus, both the first GUI element and the second GUI element are bound to the same control. The bindings between the GUI elements and the control cause the specific interactions recited in the claim. In particular, the binding between the second GUI element and the control causes the control to automatically determine the steps in the test executive sequence in response to the user input received to the second GUI element during execution of the run-time operator interface application. The binding between the first GUI element and the control causes the first GUI element to automatically display the at least a subset of the steps in response to the user input received to the second GUI element during execution of the run-time operator interface application.

In the rejection of claim 82 the Examiner cites the same portions of Grey already discussed above. However, Grey does not teach the specific combination of limitations recited in claim 82, in combination with the other recited limitations of claim 82. In particular, there is no teaching regarding the bindings configured between the GUI elements and the control and the recited functionality which is caused by the bindings.

Appellant thus respectfully submits that claim 82 is separately patentable over Grey and Stutz for at least the reasons set forth above.

Dependent Claim 85

Claim 85 recites the further limitations of, "wherein the test executive sequence is associated with a test executive environment". With respect to this limitation, the Examiner states:

> (col. 13, lines 32-33 "The TestStand Test Executive <u>Engine</u> 220 is used for creating, editing, executing, and debugging sequences.");

Claim 85 further recites, "wherein the control is operable to call the test executive environment during execution of the run-time operator interface application to determine the steps in the test executive sequence." With respect to this limitation, the Examiner states:

> (col. 13, lines 39-41 "The user can call the <u>Engine API</u> from any programming environment")

Thus, the Examiner has referred to Grey's test executive <u>engine</u> in both cases. However, claim 85 recites that <u>a control calls the test executive environment</u>. The Examiner has equated the test executive environment with the engine itself. What then does the Examiner consider to be the control? It is not at all clear. In any case, Grey does not teach a <u>control which calls the test executive engine</u> (which the Examiner has equated with the recited "test executive environment") <u>and</u> where the control includes pre-existing first functionality for determining the steps in the test executive sequence, as recited in claims 85 and 76.

Appellant thus respectfully submits that claim 85 is separately patentable over Grey and Stutz for at least the reasons set forth above.

<u>Dependent Claim 91</u>

Dependent claim 91 is indicated as being rejected in the listing of claims in the Office Action. However, the Examiner's remarks do not address claim 91, and no rationale is given for its rejection. Appellant thus submits that claim 91 does not stand properly rejected.

Appellant further notes that claim 91 recites:

> installing the application development environment on a computer system;
> installing the control on the computer system <u>**after**</u> said installing the application development environment on the computer system;

Grey and Stutz do not teach installing the control on the computer system <u>after</u> installing the application development environment on the computer system.

Appellant thus respectfully submits that claim 91 is separately patentable over Grey and Stutz for at least the reasons set forth above.

<u>Independent Claims 94 and 95</u>

The Examiner rejects claim 94 for similar reasons as the rejection of claim 76, stating:

> "Claim 94 recites limitations similar to those addressed in the rejection of claim 76 with the exception that…"

Appellant traverses the rejection of claim 94 for similar reasons as set forth above with respect to claim 76. In particular, Appellant submits that the Examiner has not established a proper motivation to combine Stutz with Grey, as argued above.

Furthermore, Appellant notes that claim 94 recites in pertinent part:

> including a control in the run-time operator interface application in response to user input, wherein the control includes pre-existing first functionality for generating a report summarizing one or more results of execution of the test executive sequence;
>
> configuring a binding between the GUI element and the control, wherein configuring the binding enables the GUI element to automatically display the report in response to the control generating the report during execution of the run-time operator interface application;

With respect to these limitations the Examiner cites Col. 8, lines 9-10; Col. 56, lines 48-50; and FIG. 50. The cited portions of Grey relate generally to step results being automatically collected during the execution of a test executive sequence. Grey's test executive system includes functionality for collecting and displaying the results. However, claim 94 relates to the creation of a run-time operator interface application by a user, e.g., a user of a test executive system. In particular, the user includes both a GUI element and a control in the run-time operator interface application and <u>configures a binding between the GUI element and the control which enables the GUI element to</u>

automatically display the report in response to the control generating the report during execution of the run-time operator interface application. Thus, the user can create a run-time operator interface application which is operable to generate and display a report during the execution of the run-time operator interface application by including the GUI element and the control in the run-time operator interface application and configuring the binding between the GUI element and the control. Grey does not teach this subject matter. Furthermore, the combination of Stutz with Grey does not remedy this deficiency.

Appellant respectfully reminds the Board that, "when evaluating the scope of a claim, every limitation in the claim must be considered. Office personnel may not dissect a claimed invention into discrete elements and then evaluate the elements in isolation. Instead, the claim as a whole must be considered. See, e.g., *Diamond v. Diehr*, 450 U.S. at 18889, 209 USPQ at 9" (as quoted from the MPEP 2106, Section II, Part C) (emphasis added). When taken as a whole, Grey does not teach the subject matter which the Examiner relies on Grey to teach. Nor does Stutz remedy the deficiency of Grey's teaching.

Appellant thus respectfully submits that claim 94 is patentably distinct over Grey and Stutz for at least the reasons set forth above. Inasmuch as claim 95 recites similar limitations as those of claim 94, Appellant submits that claim 95 is also patentably distinct for similar reasons.

Independent Claims 96 and 113

The Examiner rejects claim 96 for similar reasons as the rejection of claim 76, stating:

> "Claims 96-112 recite limitations similar to those addressed in the rejection of claims 76-93 with the exception that…"

Appellant traverses the rejection of claim 96 for similar reasons as set forth above with respect to claim 76. In particular, Appellant submits that the Examiner has not established a proper motivation to combine Stutz with Grey, as argued above.

Furthermore, Appellant notes that claim 96 recites in pertinent part:

including a GUI element in a graphical user interface of the run-time operator interface application in response to user input;

including a control in the run-time operator interface application in response to user input, wherein the control includes pre-existing first functionality for invoking execution of the test executive sequence;

configuring a binding between the GUI element and the control, wherein configuring the binding enables the control to automatically invoke execution of the test executive sequence in response to user input received to the GUI element during execution of the run-time operator interface application; and

executing the run-time operator interface application, wherein said executing comprises displaying the GUI element in the graphical user interface of the run-time operator interface application and receiving user input to the GUI element during execution of the run-time operator interface application, wherein the binding between the GUI element and the control causes the control to automatically invoke execution of the test execution sequence in response to the user input to the GUI element.

With respect to these limitations the Examiner cites Grey at Col. 23, lines 35-39. As discussed above, Grey's test executive system includes a sequence editor with a graphical user interface that enables a user to create a test executive sequence. The cited portion of Grey teaches that, "The user can start an execution in the sequence editor by selecting the Run <SequenceName> item or one of the process model entry points from the Execute menu." Thus, the sequence editor, which is provided to the user of the test executive system, includes a Run item that the user can select to start execution of the sequence. However, Grey does not teach the capability for a user to create his own run-time operator interface application by including both a GUI element and a control in the run-time operator interface application and configuring a binding between the GUI element and the control such that, when the run-time operator interface application is executed, the binding between the GUI element and the control causes the control to automatically invoke execution of the test executive sequence in response to the user input to the GUI element. Grey's cited teaching of the user starting execution of the test executive sequence refers to the user starting it from within the provided sequence editor and does not refer to the user creating a run-time operator interface application, and does not refer to the execution being started from within a user-created run-time operator interface application. Furthermore, the combination of Stutz with Grey does not remedy this deficiency.

Appellant respectfully reminds the Board that, "when evaluating the scope of a claim, every limitation in the claim must be considered. Office personnel may not dissect a claimed invention into discrete elements and then evaluate the elements in isolation. Instead, the claim as a whole must be considered. See, e.g., *Diamond v. Diehr*, 450 U.S. at 18889, 209 USPQ at 9" (as quoted from the MPEP 2106, Section II, Part C) (emphasis added). When taken as a whole, Grey does not teach the subject matter which the Examiner relies on Grey to teach. Nor does Stutz remedy the deficiency of Grey's teaching.

Appellant thus respectfully submits that claim 96, and the claims dependent thereon, are patentably distinct over Grey and Stutz for at least the reasons set forth above.

Independent 113 recites similar limitations as claim 96 except that it refers to a control for inclusion in a user-created run-time operator interface application which includes pre-existing first functionality for stopping execution of the test executive sequence. Grey teaches that at Col. 24, lines 1-4 that, "The menus in the sequence editor 212 and run-time operator interfaces 202 have commands that allow the user to stop execution before the execution has completed normally." Again, Grey is referring to menus in the sequence editor or in certain default run-time operator interfaces that are provided with the test executive system. However, the subject matter of claim 113 relates to a user-created run-time operator interface application. More particular, claim 113 relates to the capability for a user to create his own run-time operator interface application by including both a GUI element and a control in the run-time operator interface application and configuring a binding between the GUI element and the control such that, when the run-time operator interface application is executed, the binding between the GUI element and the control causes the control to automatically stop execution of the test executive sequence in response to the user input to the GUI element. Taken as a whole, Grey does not teach this subject matter. Nor does Stutz's teaching regarding object connections remedy this deficiency.

Dependent Claim 97

Claim 97 recites further limitations which are similar to those recited in independent claim 76 and discussed above. Appellant traverses the rejection of claim 97 for similar reasons as set forth above with respect to claim 76.

Dependent Claim 98

Claim 98 recites the further limitations of:

> wherein the control is a first control;
>
> wherein the method further comprises including a second control in the run-time operator interface application in response to user input, wherein the second control includes pre-existing second functionality;
>
> wherein said <u>configuring the binding between the GUI element and the first control also enables the first control to invoke the second control to perform the second functionality</u>.

The Examiner gives no rationale for rejecting claim 98 and does not address the specific recited limitation of, "wherein said configuring the binding between the GUI element and the first control also enables the first control to invoke the second control to perform the second functionality." Appellant submits that the cited references do not teach this limitation in combination with the other recited limitations.

Dependent Claim 102

Claim 102 recites similar limitations as claim 85 discussed above. Appellant respectfully submits that claim 102 is separately patentable over Grey and Stutz for reasons similar to those discussed with reference to claim 85.

Independent Claim 114

The Examiner rejects claim 114 for similar reasons as the rejection of claim 76, stating:

> "Claim 114 recites limitations similar to those addressed in the rejection of claim 76 with the exception that..."

Appellant traverses the rejection of claim 114 for similar reasons as set forth above with respect to claim 76. In particular, Appellant submits that the Examiner has not established a proper motivation to combine Stutz with Grey, as argued above.

Furthermore, Appellant notes that claim 96 recites in pertinent part:

> including a first GUI element in a graphical user interface of the run-time operator interface application in response to user input, wherein the first GUI element is operable to receive user input during execution of the run-time operator interface application;
>
> including a second GUI element in the graphical user interface of the run-time operator interface application in response to user input, wherein the second GUI element is operable to display output during execution of the run-time operator interface application;
>
> including a control in the run-time operator interface application in response to user input, wherein the control includes pre-existing first functionality for selecting the test executive sequence and pre-existing second functionality for displaying steps in the test executive sequence;
>
> configuring a first binding between the first GUI element and the control, wherein the first binding enables the control to automatically invoke a dialog box enabling a user to select the test executive sequence in response to user input received to the first GUI element during execution of the run-time operator interface application;
>
> configuring a second binding between the second GUI element and the control, wherein the second binding enables the second GUI element to automatically display the steps in the test executive sequence in response to the user selecting the test executive sequence during execution of the run-time operator interface application;

Grey's sequence editor includes the built-in capability for a user to load a sequence file and view the steps of the sequence. However, claim 114 relates to the creation of a run-time operator interface application by a user, e.g., a user of a test executive system. In particular, the user includes a first GUI element, a second GUI element, and a control in the run-time operator interface application. A first binding between the first GUI element and the control is created, where the first binding enables the control to automatically invoke a dialog box enabling a user to select the test executive sequence in response to user input received to the first GUI element during execution of the run-time operator interface application. Also, a second binding between the second GUI element and the (same) control is created, where the second binding enables the second GUI element to automatically display the steps in the test executive sequence in response to the user selecting the test executive sequence during execution of the run-time operator interface application. This specific combination of limitations, when considered as a whole in combination with the other recited limitations of claim 114, is not taught by Grey and Stutz, taken either singly or in combination.

<u>Section 103 Rejections (Grey in view of Stutz and Carter)</u>

Claim 91 stands rejected under 35 U.S.C. 103(a) as being unpatentable over Grey in view of Stutz, and further in view of Carter.

Claim 91 depends on claim 89, which depends on claim 76. As argued above, Appellant respectfully submits that claim 76 is patentably distinct over the cited art. Appellant thus respectfully submits that claim 91, which is dependent thereon, is also patentably distinct for at least this reason.

VIII. CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejection of the claims was erroneous, and reversal of the decision is respectfully requested.

The Commissioner is authorized to charge the appeal brief fee of $540.00 and any other fees that may be due to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5150-77400/JCH.

Respectfully submitted,

/Jeffrey C. Hood/
Jeffrey C. Hood Reg. #35198
ATTORNEY FOR APPLICANT(S)

Meyertons Hood Kivlin Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800

Date: 2008-12-15 JCH/JLB

IX. CLAIMS APPENDIX

The following lists the claims as incorporating entered amendments, and as on appeal.

1-75. (Canceled)

76. (Previously Presented) A computer-implemented method for displaying information regarding a test executive sequence, wherein the test executive sequence includes a plurality of steps, the method comprising:

including a GUI element in a graphical user interface of a run-time operator interface application in response to user input, wherein the GUI element is operable to display information;

including a control in the run-time operator interface application in response to user input, wherein the control includes pre-existing first functionality for determining the steps in the test executive sequence;

configuring a binding between the GUI element and the control, wherein configuring the binding enables the GUI element to automatically display at least a subset of the steps in the test executive sequence in response to the control determining the steps in the test executive sequence during execution of the run-time operator interface application; and

executing the run-time operator interface application, wherein said executing comprises the control executing to automatically determine the steps in the test executive sequence, wherein the binding between the GUI element and the control causes the GUI element to automatically display at least a subset of the steps in response to the control determining the steps, wherein the GUI element displays the at least a subset of the steps in the graphical user interface of the run-time operator interface application during execution of the run-time operator interface application.

77. (Previously Presented) The method of claim 76,

wherein the control also includes pre-existing functionality for formatting the at least a subset of the steps in the test executive sequence into a formatted list;

wherein the GUI element automatically displaying the at least a subset of the steps comprises the GUI element automatically displaying the formatted list.

78. (Previously Presented) The method of claim 77,

wherein in performing said formatting the at least a subset of the steps in the test executive sequence into the formatted list, the control is operable to:

determine information regarding each step of the at least a subset of the steps in the test executive sequence; and

format the information for display in the GUI element;

wherein the formatted list includes the formatted information for each of the steps in the at least a subset of the steps.

79. (Previously Presented) The method of claim 76,

wherein the test executive sequence is stored in a sequence file;

wherein in automatically determining the steps in the test executive sequence, the control is operable to automatically obtain information from the sequence file regarding the test executive sequence and determine the steps based on the information obtained from the sequence file.

80. (Previously Presented) The method of claim 76, further comprising:

configuring the control in response to configuration user input after said including the control in the run-time operator interface application, wherein the configuration user input specifies an appearance for the displayed steps, wherein configuring the control enables the control to cause the steps to be displayed in the GUI element with the specified appearance.

81. (Previously Presented) The method of claim 80,

wherein the configuration user input specifies one or more properties regarding a plurality of columns to display in the GUI element;

wherein configuring the control enables the control to cause information for each displayed step to be displayed in the GUI element in the plurality of columns according to the one or more specified properties.

82. (Previously Presented) The method of claim 76,

wherein the GUI element comprises a first GUI element;

wherein the method further comprises:

including a second GUI element in the run-time operator interface application in response to user input; and

configuring a binding between the second GUI element and the control;

wherein executing the run-time operator interface application comprises the second GUI element receiving user input during execution of the run-time operator interface application;

wherein the binding between the second GUI element and the control causes the control to automatically determine the steps in the test executive sequence in response to the user input received to the second GUI element during execution of the run-time operator interface application;

wherein the binding between the first GUI element and the control causes the first GUI element to automatically display the at least a subset of the steps in response to the user input received to the second GUI element during execution of the run-time operator interface application.

83. (Previously Presented) The method of claim 76,

wherein said including the control in the run-time operator interface application enables a user to configure the run-time operator interface application to automatically determine the steps in the test executive sequence without requiring the user to create program code to determine the steps in the test executive sequence;

wherein said configuring the binding between the GUI element and the control enables the user to configure the run-time operator interface application to automatically display the at least a subset of the steps in the test executive sequence without requiring the user to create program code for displaying the steps in the test executive sequence.

84. (Previously Presented) The method of claim 76,

wherein the test executive sequence is operable to perform one or more tests on one or more units under test (UUTs).

85. (Previously Presented) The method of claim 76,

wherein the test executive sequence is associated with a test executive environment;

wherein the control is operable to call the test executive environment during execution of the run-time operator interface application to determine the steps in the test executive sequence.

86. (Previously Presented) The method of claim 76,

wherein the control comprises a software component constructed in accordance with an ActiveX^{TM} specification.

87. (Canceled)

88. (Previously Presented) The method of claim 76,

wherein the control does not appear on the graphical user interface of the run-time operator interface application during execution of the run-time operator interface application.

89. (Previously Presented) The method of claim 76,

wherein the control is a pre-existing control provided by an application development environment used to create the run-time operator interface application.

90. (Previously Presented) The method of claim 89, further comprising:

installing the application development environment on a computer system, wherein said installing the application development environment on the computer system comprises installing the control on the computer system.

91. (Previously Presented) The method of claim 89, further comprising:

installing the application development environment on a computer system;

installing the control on the computer system after said installing the application development environment on the computer system;

wherein said installing the control on the computer system enables the application development environment to provide a user with access to the control.

92. (Previously Presented) The method of claim 76,

wherein said configuring the binding between the GUI element and the control comprises performing one or more calls to bind the GUI element to the control during execution of the run-time operator interface application.

93. (Previously Presented) The method of claim 76,

wherein said configuring the binding between the GUI element and the control is performed in response to receiving user input to a graphical user interface to specify the binding between the GUI element and the control.

94. (Previously Presented) A computer-implemented method for displaying a report for a test executive sequence execution, the method comprising:

including a GUI element in a graphical user interface of a run-time operator interface application in response to user input, wherein the GUI element is operable to display information;

including a control in the run-time operator interface application in response to user input, wherein the control includes pre-existing first functionality for generating a report summarizing one or more results of execution of the test executive sequence;

configuring a binding between the GUI element and the control, wherein configuring the binding enables the GUI element to automatically display the report in response to the control generating the report during execution of the run-time operator interface application;

configuring the run-time operator interface application to invoke execution of the test executive sequence; and

executing the run-time operator interface application, wherein the run-time operator interface application executes to invoke execution of the test executive sequence, wherein the execution of the test executive sequence produces the one or more results, wherein the control automatically generates the report summarizing the one or more results of the execution of the test executive sequence in response to the execution of the test executive sequence, wherein the binding between the GUI element and the control causes the report to be automatically displayed by the GUI element in response to the control generating the report, wherein the GUI element displays the report in the graphical user interface of the run-time operator interface application during execution of the run-time operator interface application.

95. (Previously Presented) A computer-implemented method for displaying information for a test executive sequence execution, the method comprising:

including a GUI element in a graphical user interface of a run-time operator interface application in response to user input, wherein the GUI element is operable to display information;

including a control in the run-time operator interface application in response to user input, wherein the control includes pre-existing first functionality for generating information indicating one or more execution results for the test executive sequence;

configuring a binding between the GUI element and the control, wherein configuring the binding enables the GUI element to automatically display the information in response to the control generating the information during execution of the run-time operator interface application;

configuring the run-time operator interface application to invoke execution of the test executive sequence; and

executing the run-time operator interface application, wherein the run-time operator interface application executes to invoke execution of the test executive sequence, wherein the execution of the test executive sequence produces the one or more execution results, wherein the control automatically generates the information indicating the one or more execution results in response to the execution of the test executive sequence, wherein the binding between the GUI element and the control causes the information indicating the one or more execution results to be automatically displayed by the GUI element in response to the control generating the information, wherein the GUI element displays the information in the graphical user interface of the run-time operator interface application during execution of the run-time operator interface application during execution of the run-time operator interface application.

96. (Previously Presented) A computer-implemented method for creating a run-time operator interface application for controlling execution of a test executive sequence, the method comprising:

including a GUI element in a graphical user interface of the run-time operator interface application in response to user input;

including a control in the run-time operator interface application in response to user input, wherein the control includes pre-existing first functionality for invoking execution of the test executive sequence;

configuring a binding between the GUI element and the control, wherein configuring the binding enables the control to automatically invoke execution of the test executive sequence in response to user input received to the GUI element during execution of the run-time operator interface application; and

executing the run-time operator interface application, wherein said executing comprises displaying the GUI element in the graphical user interface of the run-time operator interface application and receiving user input to the GUI element during execution of the run-time operator interface application, wherein the binding between the GUI element and the control causes the control to automatically invoke execution of the test execution sequence in response to the user input to the GUI element.

97. (Previously Presented) The method of claim 96,

wherein the control also includes pre-existing second functionality for determining steps in the test executive sequence;

wherein the GUI element comprises a first GUI element;

wherein the method further comprises:

including a second GUI element in the graphical user interface of the run-time operator interface application in response to user input; and

configuring a binding between the second GUI element and the control, wherein configuring the binding between the second GUI element and the control enables the second GUI element to automatically display at least a subset of the steps in the test executive sequence in response to the control determining the steps in the test executive sequence during execution of the run-time operator interface application;

wherein said executing the run-time operator interface application further comprises the control automatically determining the steps in the test executive sequence, wherein the binding between the second GUI element and the control causes the second GUI element to automatically display at least a subset of the steps in response to the control determining the steps, wherein the second GUI element displays the at least a subset of the steps in the graphical user interface of the run-time operator interface application.

98. (Previously Presented) The method of claim 96,

wherein the control is a first control;

wherein the method further comprises including a second control in the run-time operator interface application in response to user input, wherein the second control includes pre-existing second functionality;

wherein said configuring the binding between the GUI element and the first control also enables the first control to invoke the second control to perform the second functionality.

99. (Previously Presented) The method of claim 96,

wherein said configuring the binding between the GUI element and the control enables a user to configure the run-time operator interface application to invoke execution of the test executive sequence in response to user input received to the GUI element without requiring the user to write program code to program the run-time operator interface application to respond to user input received to the GUI element.

100. (Previously Presented) The method of claim 96,

wherein said configuring the binding between the GUI element and the control enables a user to configure the run-time operator interface application to invoke execution of the test executive sequence in response to user input received to the GUI element without requiring the user to create program code to program the run-time operator interface to invoke execution of the test executive sequence.

101. (Previously Presented) The method of claim 96,

wherein the control is operable to invoke execution of the test executive sequence to perform one or more tests on one or more units under test (UUTs).

102. (Previously Presented) The method of claim 96,

wherein the test executive sequence is associated with a test executive environment;

wherein the control is operable to automatically call the test executive environment during execution of the run-time operator interface application to invoke execution of the test executive sequence.

103. (Previously Presented) The method of claim 96,

wherein the control comprises a software component constructed in accordance with an ActiveX^{TM} specification.

104. (Previously Presented) The method of claim 96,

wherein the GUI element appears on a graphical user interface of the run-time operator interface application during execution of the run-time operator interface application;

wherein the GUI element is operable to receive user input to the graphical user interface during execution of the run-time operator interface application.

105. (Previously Presented) The method of claim 104,

wherein the control does not appear on the graphical user interface of the run-time operator interface application during execution of the run-time operator interface application.

106. (Previously Presented) The method of claim 96,

wherein the control is a pre-existing control provided by an application development environment used to create the run-time operator interface application.

107. (Previously Presented) The method of claim 96,

wherein said configuring the binding between the GUI element and the control is performed in response to receiving user input to a graphical user interface to specify the binding between the GUI element and the control.

108. (Canceled)

109. (Previously Presented) The method of claim 96,

wherein said including the GUI element in the run-time operator interface application in response to user input comprises displaying the GUI element on a graphical user interface of the run-time operator interface application in response to user input.

110. (Previously Presented) The method of claim 96, further comprising:

configuring one or more properties of the control in response to user input after said configuring the binding between the GUI element and the control.

111. (Previously Presented) The method of claim 110, further comprising:

displaying a property panel for configuring the control; and

receiving user input to the property panel to configure the one or more properties of the control.

112. (Previously Presented) The method of claim 96,

wherein the GUI element comprises one or more of:

a button;

a text input element;

a check box;

a selection ring.

113. (Previously Presented) A computer-implemented method for creating a run-time operator interface application for controlling execution of a test executive sequence, the method comprising:

including a GUI element in a graphical user interface of the run-time operator interface application in response to user input;

including a control in the run-time operator interface application in response to user input, wherein the control includes pre-existing first functionality for stopping execution of the test executive sequence;

configuring a binding between the GUI element and the control, wherein configuring the binding enables the control to automatically stop execution of the test executive sequence in response to user input received to the GUI element during execution of the run-time operator interface application; and

executing the run-time operator interface application, wherein said executing comprises displaying the GUI element in the graphical user interface of the run-time operator interface application and receiving user input to the GUI element during execution of the run-time operator interface application, wherein the binding between the GUI element and the control causes the control to automatically stop execution of the test execution sequence in response to the user input to the GUI element.

114. (Previously Presented) A computer-implemented method for creating a run-time operator interface application for controlling execution of a test executive sequence, the method comprising:

including a first GUI element in a graphical user interface of the run-time operator interface application in response to user input, wherein the first GUI element is operable to receive user input during execution of the run-time operator interface application;

including a second GUI element in the graphical user interface of the run-time operator interface application in response to user input, wherein the second GUI element is operable to display output during execution of the run-time operator interface application;

including a control in the run-time operator interface application in response to user input, wherein the control includes pre-existing first functionality for selecting the test executive sequence and pre-existing second functionality for displaying steps in the test executive sequence;

configuring a first binding between the first GUI element and the control, wherein the first binding enables the control to automatically invoke a dialog box enabling a user to select the test executive sequence in response to user input received to the first GUI element during execution of the run-time operator interface application;

configuring a second binding between the second GUI element and the control, wherein the second binding enables the second GUI element to automatically display the steps in the test executive sequence in response to the user selecting the test executive sequence during execution of the run-time operator interface application; and

executing the run-time operator interface application, wherein said executing comprises:

displaying the first GUI element and the second GUI element in the graphical user interface of the run-time operator interface application;

receiving user input to the first GUI element;

the control automatically invoking a dialog box enabling a user to select the test executive sequence in response to the user input to the first GUI element;

receiving user input selecting the test executive sequence via the dialog box; and

the second GUI element automatically displaying the steps in the test executive sequence in response to the user input selecting the test executive sequence, wherein the second GUI element displays the steps in the graphical user interface of the run-time operator interface application during execution of the run-time operator interface application.

X. <u>EVIDENCE APPENDIX</u>

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

XI. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.